

UNITED STATES PATENT APPLICATION

for

SOFTWARE MODEM ARCHITECTURE

INVENTOR:

**Dennis Kwan**

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP  
12400 WILSHIRE BOULEVARD  
SEVENTH FLOOR  
LOS ANGELES, CALIFORNIA 90025  
(408) 720-8598

Attorney's Docket No. 04939.P001

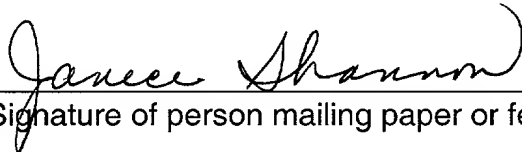
"Express Mail" mailing label number: EL617178264US

Date of Deposit May 8, 2001

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Janece Shannon

(typed or printed name of person mailing paper or fee)



(Signature of person mailing paper or fee)

## PRIORITY

[0001] This application claims the benefit of U.S. Provisional Application No. 60/202,734, filed May 8, 2000.

## BACKGROUND OF THE INVENTION

### Field of the Invention

[0002] This invention relates generally to communication systems. More particularly, the invention relates to an improved architecture for implementing communication protocols such as the “Bluetooth” wireless protocol.

### Description of the Related Art

[0003] Various combinations of hardware, firmware and software may be used to implement wireless and terrestrial communication protocol stacks. For example, referring to **Figure 1**, the “Bluetooth” specification is comprised of several different protocol layers including a radio frequency (“RF”) layer 160, a baseband layer (“BB”) 150, a link control layer (“LC”) 140, a link manager layer (“LM”) 130, a logical link control and adaptation protocol layer (“L2CAP”), and a serial line emulation layer (“RFCOMM”). The functionality of each of these layers (as well as additional Bluetooth protocol layers) is described in detail in *Bluetooth Protocol Architecture, Version 1.0* (August 25, 1999) (“*Bluetooth Protocol Architecture*”), which can be found at “<http://www.Bluetooth.com>.”



performs an electronic operation as fast as it is possible to do so, providing that the circuit is efficiently designed.

[0006] The Bluetooth IC in **Figure 1** communicates with a host processor environment 105 through a host interface 107. The host processor environment 105 is typically comprised of a general purpose CPU (e.g., an Intel Pentium®-class processor) and software executed by the CPU (e.g., an operating system with an application programming interface (“API”) and one or more application programs). As shown, in a typical configuration the host processor environment 105 is configured to support only the upper layers of the protocol stack, RF Comm 110, and L2CAP 120.

## SUMMARY OF THE INVENTION

[0007] A software modem is described comprising: a finite state machine having a plurality of states interconnected through a plurality of events, wherein certain states and events in said plurality are implemented in software and other states and events in said plurality are implemented in hardware; and a scheduler communicatively coupled to the finite state machine and being programmable with one or more parameters defining scheduled operations to be performed by the scheduler, wherein the finite state machine is configured to select one or more of said parameters to be used by said scheduler upon transition by said finite state machine from a first state to a second state.

TCW 04939.P001

## BRIEF DESCRIPTION OF THE DRAWINGS

[0008] A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

[0009] **FIG. 1** illustrates a typical allocation of a Bluetooth protocol stack between a host processing environment and a Bluetooth IC.

[0010] **FIG. 2** illustrates a software modem architecture according to one embodiment of the invention.

[0011] **FIG. 3** illustrates a series of states and events allocated between hardware and software according to one embodiment of the invention.

[0012] **FIG. 4** illustrates an exemplary look-up table which may be implemented in one embodiment of the invention.

[0013] **FIG. 5** illustrates a programmable finite state machine (“PFSM”) engine implemented in one embodiment of the invention.

[0014] **FIG. 6** illustrates a scheduler module implemented in one embodiment of the invention.

[0015] **FIG. 7a** illustrates an exemplary timing diagram according to one embodiment of the invention.

[0016] **FIG. 7b** illustrates another exemplary timing diagram according to one embodiment of the invention.

[0017] **FIG. 8** illustrates a master page operation implemented according to one embodiment of the invention.

[0018] **FIG. 9** illustrates protocol stack allocation according to one embodiment of the invention.

[0019] **FIG. 10** illustrates a packet processor according to one embodiment of the invention.

TCW 4/13/00

## DETAILED DESCRIPTION

[0020] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. For example, although embodiments of the invention are described below in the context of the "Bluetooth" standard, it should be noted that the underlying principles of the invention are not limited to any particular communication standard. For example, features of the invention may be performed in virtually any wireless or wireline network environment. Moreover, in some instances, well-known structures and devices are shown in block diagram form to avoid obscuring the principles of the invention.

## INTRODUCTION

[0021] A system and method is described below which may be applied to a multitude of wireless and wireline protocols which, in general, rely on very fast, hard real-time responses within the physical or medium access control ("MAC") protocol layers. While this type of system and method is not always required for wireline modems such as V.90 (56k) dialup modems, it is necessary for a Time Division Duplexing ("TDD") wireless protocol such as Bluetooth. One embodiment of the invention implements a software based modem on a host processor that does not respond to real-time events fast enough due, for example, to a slow link between the target hardware and the host processor (e.g.

serial RS232), or the host processor running an OS that is non-real time (e.g. Windows). In addition, according to embodiments of the invention described herein, the software-based modem may be configured to process a larger portion of the protocol stack.

**[0022]** Moreover, embodiments of the invention described below provide increased flexibility by logically partitioning a protocol finite state machine (“FSM”) between hardware and software. This property, combined with the programmable nature of the FSM, allows one to dynamically “fine-tune” the performance of the modem according to the capability of the host CPU. For example, with a relatively high-powered processor such as a 1GHz CPU Pentium a larger number of slave devices may be handled in a piconet (i.e., when compared with slower CPU).

#### **EMBODIMENTS OF THE INVENTION**

**[0023]** Although ASICs improve performance over general-purpose CPUs for the reasons set forth above, they incur additional costs to the overall product when compared with software that may be executed by a general-purpose CPU and that may already exist for that product. As such, in some situations (e.g., for low-speed applications or applications which require only partial real-time support, or in situations where the host processor is fast enough to handle the necessary high-speed processing) it may be more efficient to offload the lower level protocol layers 130, 140, 150, and 160 from the Bluetooth IC to the host processing environment 105. This configuration will reduce the gate hardware

requirement for the Bluetooth IC and thereby reduce overall product costs for adding Bluetooth capability to electronic devices that already require a host CPU.

**[0024]** One embodiment of a software modem, illustrated in **Figure 2**, is comprised of a finite state machine (“FSM”) engine 230; a scheduler unit 220 including a priority control unit 225; a hop sequencer 240 communicatively coupled to a phase locked loop (“PLL”) unit through a PLL interface 270; one or more Bluetooth clocks 210 feeding a clock signal to the scheduler unit 220 and the hop sequencer 240; and an RF unit comprised of a packet processor 260 for receiving/ transmitting (“TxD, RxD”) incoming/outgoing data packets and a control unit 250 for controlling (e.g., enabling/disabling) the packet processor 260.

### ***FSM Engine***

**[0025]** The FSM engine 230 of one embodiment provides a simple, efficient programming model for the baseband processor which allows baseband functions to be implemented using a limited number of gates in the baseband IC (and therefore provides a smaller IC footprint). More particularly, referring to **Figures 3** and **4**, which illustrate a state diagram 300 and a state look-up table 400 respectively, the FSM engine 230 is comprised of a series of baseband processor states 301-307 connected through a series of baseband processor events 310-318. In operation, when the baseband processor is operating in one state, e.g., state 302, for example, and detects an event 311, the processor will transition to state 305 and wait for the next event. In addition, as indicated in the

look-up table 400, one or more baseband processor actions (e.g., Action B) may also be associated with a state/event pair.

**[0026]** As an example, at state 301 the baseband processor may listen for an incoming data packet; event 310 may be the receipt by the processor of an incoming data packet; and action 'A' taken by the processor may be a data packet transmission by the processor in response to the received data packet. Various other states, events and actions may be employed consistent with the underlying principles of the invention.

**[0027]** For time-critical applications such as real-time audio/video transmissions and other types of high-speed data applications, transitions from one state to another must be dealt with very quickly. By contrast, other types of applications such as wireless input devices (i.e., keyboards, mice, game controllers . . . etc) do not require the same level of high-speed event transitioning.

**[0028]** In addition, in a communication protocol stack, some kind of FSM logic may be implemented in almost every layer to provide the required handshake and message exchange between two devices. As described previously, the real-timeliness of the transitions between states generally decreases as one goes up the protocol layers. At the same time, within each layer the response time requirements between different states also vary significantly depending on the particular design of the protocol. Accordingly, as illustrated in **Figure 3**, for these different types of requirements, portions of the FSM engine 230 may be

implemented using software 320 as well as hardware 330. In addition, it should be noted that even applications which require real-time/high-speed functionality may be implemented as described herein if the host processor is powerful enough. In one embodiment, the software component 320 is executed in the host processing environment 205 and the hardware component 330 is programmed in an ASIC. It should also be noted that portions of the state diagram 300 may exist at the link control ("LC") layer as well as the baseband layer.

**[0029]** If the system is configured to support the Bluetooth protocol, the protocol stack may be divided as illustrated in **Figure 9**, with the RF layer 960 and portions of the baseband layer 950 implemented in a Bluetooth IC 906 (which may be an ASIC) and the remaining layers, including certain portions of the baseband layer 150, implemented as software 320 executed in the host processor environment 905. In this embodiment, transitions between hardware states (e.g., state 302) and software states (e.g., state 305) will occur over the host interface 907.

**[0030]** As illustrated in **Figure 5**, one embodiment of an FSM engine 230 is comprised generally of a lookup table unit 400, a decode unit 540 and a latch 550. The lookup table unit 400 of one embodiment is comprised of a memory (not shown) for storing a lookup table such as the one illustrated in **Figure 4**. The memory may be a volatile memory (e.g., random access memory or "RAM") or a non-volatile memory (e.g., an electrically erasable programmable read only

memory or “EEPROM”; a programmable read only memory or “PROM”, etc), depending on the particular embodiment. It should be noted, however, that the underlying principles of the invention may be implemented using virtually any type of memory capable of storing look-up table data. New look-up table data may be generated to support new applications and loaded/programmed into the memory in the look-up table unit 400 via the Load data path 510. The look-up table data in one embodiment is transmitted from the host processor environment 205 (via the load data path 510).

**[0031]** The FSM engine 230 of one embodiment also includes an address input 401 and a data output 402. New events 530 and the current state of the FSM engine 230 (stored in latch 550) are combined to form a look-up table address which is used by the FSM engine 230 to identify the next state 560 to which the FSM engine 230 will transition and/or the next action 520 which the FSM engine 230 will perform. The next state 560 may either be programmed in software or in hardware as described above. If programmed in software, the next state signal will be transmitted over the host interface 907 and processed in the host processing environment 905.

**[0032]** Upon transition to a new state, an action code 520 may be generated by the FSM engine 230. The action code is received by the decode logic unit 540, which produces a predefined action signal in response (e.g., such as transmit enable “TxEN” or receive enable “RxEN” as illustrated in **Figure 5**). The action signals may be executed in the baseband layer 950 or, alternatively, may

be transmitted down the protocol stack to the RF layer 960 or up the protocol stack to the link control layer 940, depending on the type of action required.

### ***Scheduler***

**[0033]** As illustrated in **Figure 2**, one embodiment of the software modem also includes a scheduler unit 220 which operates in conjunction with the FSM engine 230. When implementing wireless/networking protocols (e.g., such as Bluetooth) there are typically predefined time periods during which certain actions must be taken (e.g., transmit packets, listen for packets, etc). The scheduler unit 220 provides a simple, efficient mechanism and programming model for supporting these types of timed, periodic actions.

**[0034]** As illustrated in **Figure 7a**, the host processor may offload operations to the scheduler unit 220 by specifying parameters including a start count value 710, a period 730, a repetition number 740, and one or more actions to be performed 720, 721. The start count 710 specifies a point in time when the scheduler unit 220 should begin executing a specified series of operations. The period 730 indicates the amount of time which should be allocated to perform one iteration of a particular set of actions 720, 721. For example, as illustrated in **Figure 2b**, when implemented in a Bluetooth environment, the period 730 may be defined by a specified number of 625 $\mu$ s time slots (e.g., slots 1a-10a; 1b-10b; etc), with each slot representing a different action (e.g., transmit in slot 1a, receive in slot 2a), and a specified hop frequency.

[0035] The number of periods 730 to be executed by the scheduler unit 220 may be specified by a “number of repetitions” parameter 740. This value may be set to a definite number (e.g., 10) or, alternatively, may be set to an infinite value (i.e., execute until interrupted).

[0036] One particular embodiment of a scheduler unit 220 is illustrated in **Figure 6**. This embodiment includes a set of parameter registers 610 for storing the various scheduler unit 220 parameters described above. New parameters may be loaded into the registers 610 via a load signal 612 originating from the host processor environment. Once the values are loaded, the FSM engine 230 may select a particular set of parameters via a select signal 614. In one embodiment, the select signal 614 transmitted to the scheduler unit 220 is defined by an FSM engine action 520 as described above.

[0037] In one embodiment, the scheduler 220 compares the start count value 710 to with a Bluetooth clock 670 to determine when to initiate the scheduled routine. As described above, each Bluetooth clock 670 operates on a specified frequency hop sequence. The scheduler unit 220 may identify a particular hop count within the hop sequence as the start count. Upon identification of the start count, an enable signal 660 is transmitted to the period counter 620. In one embodiment (e.g., in a Bluetooth environment) the slot clock 672, having a period of 625  $\mu$ s (i.e., a frequency of 1600 slots/sec), triggers the period counter 620. In response, the period counter 620 identifies the beginning and end of each period

730 based on the number of time slots counted (e.g., in **Figure 7b** each “period” is equal to 10 time slots).

[0038] A trigger signal 662 having a period based on the slot clock 672 period is transmitted as an input to a slot counter 630 which identifies the beginning and end of each time slot. In a Bluetooth environment, the slot counter 630 triggers on a  $\frac{1}{2}$  slot clock 674 because this is the resolution required by the Bluetooth protocol. In response to the trigger signal 662 and the  $\frac{1}{2}$  slot clock signal 630, a binary control signal 664 is transmitted to a lookup table unit 640.

[0039] In one embodiment, the lookup table unit 640 uses the control signal 664 as an address to perform a lookup operation and determine the next action 667 to be executed. For example, referring once again to **Figure 7b**, the scheduler unit 220 may generate a “transmit” command in the first slot of the period (e.g., slot 1a) based on a control signal of ‘0’ and may generate a “receive” command in the second slot of the period (e.g., slot 1b) based on a control signal of ‘1.’ It should be noted, however, that various more complex look-up addressing operations may be performed consistent with the underlying principles of the invention (e.g., using control addresses with multiple digits). New look-up tables may be loaded into a memory configured in the look-up table unit 640 via a load signal 669 and individual look-up tables may be selected by a table-select parameter 665 stored in one of the parameter registers 610.

### ***Master Page – One Example of System Operation***

**[0040]** Interaction between the FSM engine 230, the scheduler unit 220 and the host processor environment 205 will now be described with reference to **Figure 8** which illustrates a Bluetooth page operation. Briefly, the page operation is used to establish communication between a “master” Bluetooth device and one or more “slave” devices. In the embodiment illustrated in **Figure 8** the initial sequence 810 of the page operation is performed by software executed in the host processor environment 205. The master device attempts to capture the slave device by transmitting pages which include the slave device’s access code (“DAC”) on different hop channels. Since the Bluetooth clocks of the master device and the slave device are not synchronized, the master device does not know exactly when the slave device wakes up and on which hop frequency. Therefore, it transmits a train of identical DACs at different hop frequencies, listening between the transmit intervals until it receives a response from the slave.

**[0041]** The master device also selects a packet type and an active member address code (“AM\_ADDR”). The packet type identifies (among other things) the number of time slots the packet will occupy (up to five Bluetooth time slots may be occupied by a packet) and the AM\_ADDR identifies active members of the local Bluetooth network (referred to as a “piconet”) once a master-slave connection is established.

**[0042]** Thus, transmissions 801 and 802 which contain the DAC of the slave are transmitted on two different hop frequencies. In the following receive (“Rx”) slot 803, the receiver listens sequentially to two corresponding Rx hops for an incoming packet with the slave DAC. The master device continues these transmissions at various hop frequencies until it receives a response from the slave (i.e., one which contains the slave’s DAC).

**[0043]** When it receives such a transmission, the FSM engine 230 (at sequence 820) freezes the input to the hop sequencer. In addition, it transitions to a new state (a “master page response” state) and selects a new set of parameters (e.g., from the parameter registers 610) so that the scheduler unit 220 will execute the master device’s page response.

**[0044]** At sequence 830, the scheduler unit 220 executes the response by transmitting an FHS packet 804 in the next slot following the Rx event. An FHS packet 804 is a special control packet transmitted in a single time slot which contains, among other things, the Bluetooth device’s address and the clock of the master device. After the master device transmits the FHS packet 804, it waits for a second response from the slave device which acknowledges the reception of the FHS packet 804 with a packet containing the slave device’s DAC.

**[0045]** At this point (see sequence 840) the FSM engine 230 selects a particular hop sequence referred to as the “channel hopping sequence,” which uses all 79 hop channels in a random fashion. The master device may now send its first traffic packet in a hop frequency based on the new master parameters.

As illustrated in **Figure 8**, this first packet will be a POLL packet 805. The slave device is expected to respond to the POLL packet 805 with any type of data packet over the newly-generated master-slave channel. Finally, at sequence 850, the host processor loads the scheduler unit 220 with parameters for the new connection.

**[0046]** It is important to note that the apparatus and method described herein may be implemented in environments other than a physical integrated circuit ("IC"). For example, the circuitry may be incorporated into a format or machine-readable medium for use within a software tool for designing a semiconductor IC. Examples of such formats and/or media include computer readable media having a VHSIC Hardware Description Language ("VHDL") description, a Register Transfer Level ("RTL") netlist, and/or a GDSII description with suitable information corresponding to the described apparatus and method.

**[0047]** Throughout the foregoing description, for the purpose of explanation, numerous specific details were set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without some of these specific details. For example, while the embodiments described above focused on the Bluetooth protocol, the underlying principles of the invention may practiced using various other types of wireless and terrestrial protocols. Similarly, while the interaction between the scheduler unit 220, the FSM engine 230 and the host processor environment was described in the context of a master page operation, various

other operations may be performed while complying with the underlying principles of the invention. Accordingly, the scope and spirit of the invention should be judged in terms of the claims which follow.

TCW-04939-001